

## *OntoDSL*

# An Ontology-Based Framework for Domain-Specific Languages

Tobias Walter

Fernando Silva Parreiras

09.06.2009

## Motivation

- Scenario
- Requirements

## Ontology-based DSL Frameworks

- Developing DSLs
- Using DSLs

## Benefits

- Reasoning Services

## Conclusion and Discussion

## DSL User

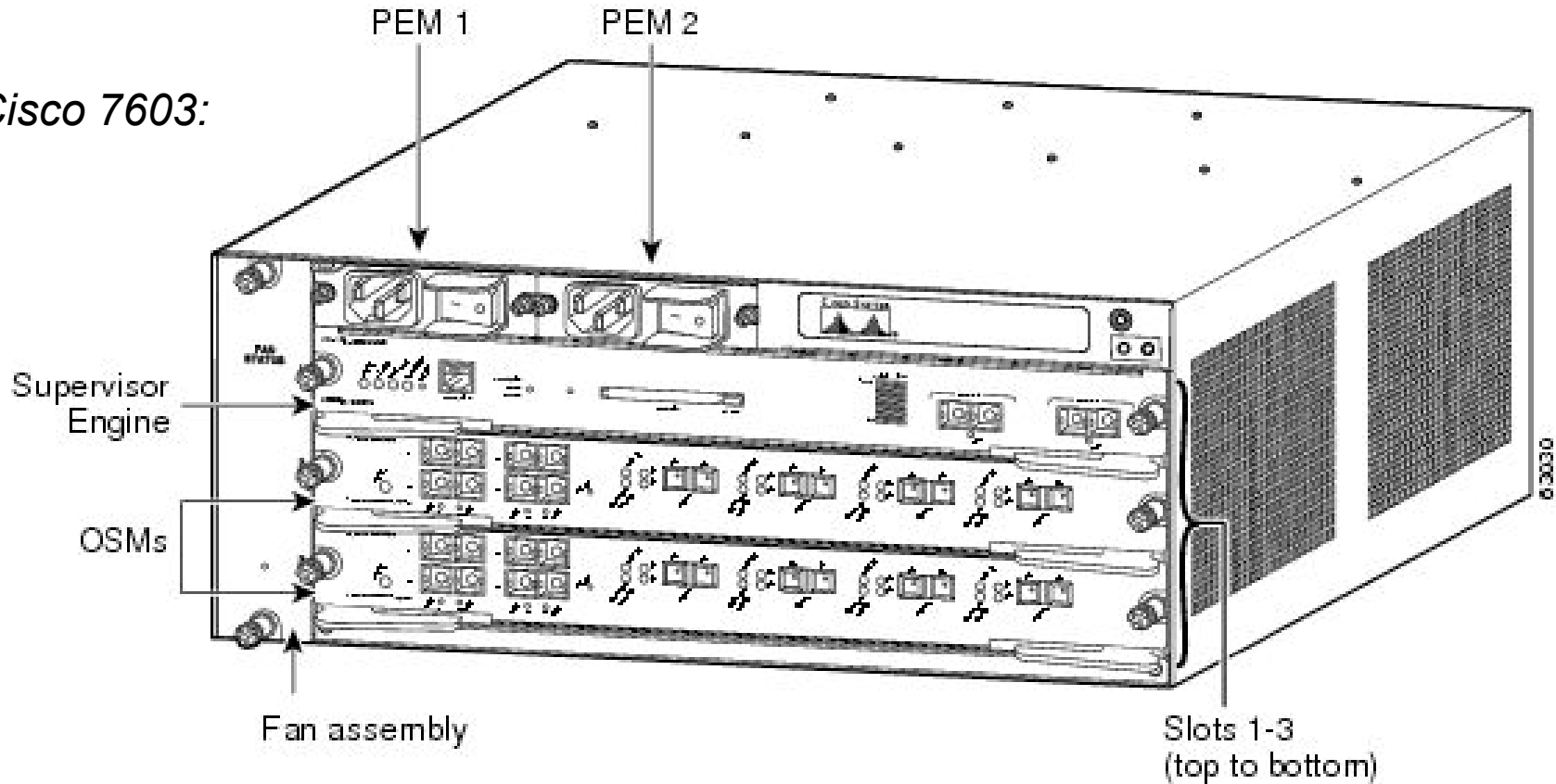
- Uses Domain Specific Language to create Domain Models
- E.g. Models are Financial Contracts (Bank Officer), Network Device Configuration (System Engineer)
- Needs Services for productively Modeling

## DSL Designer

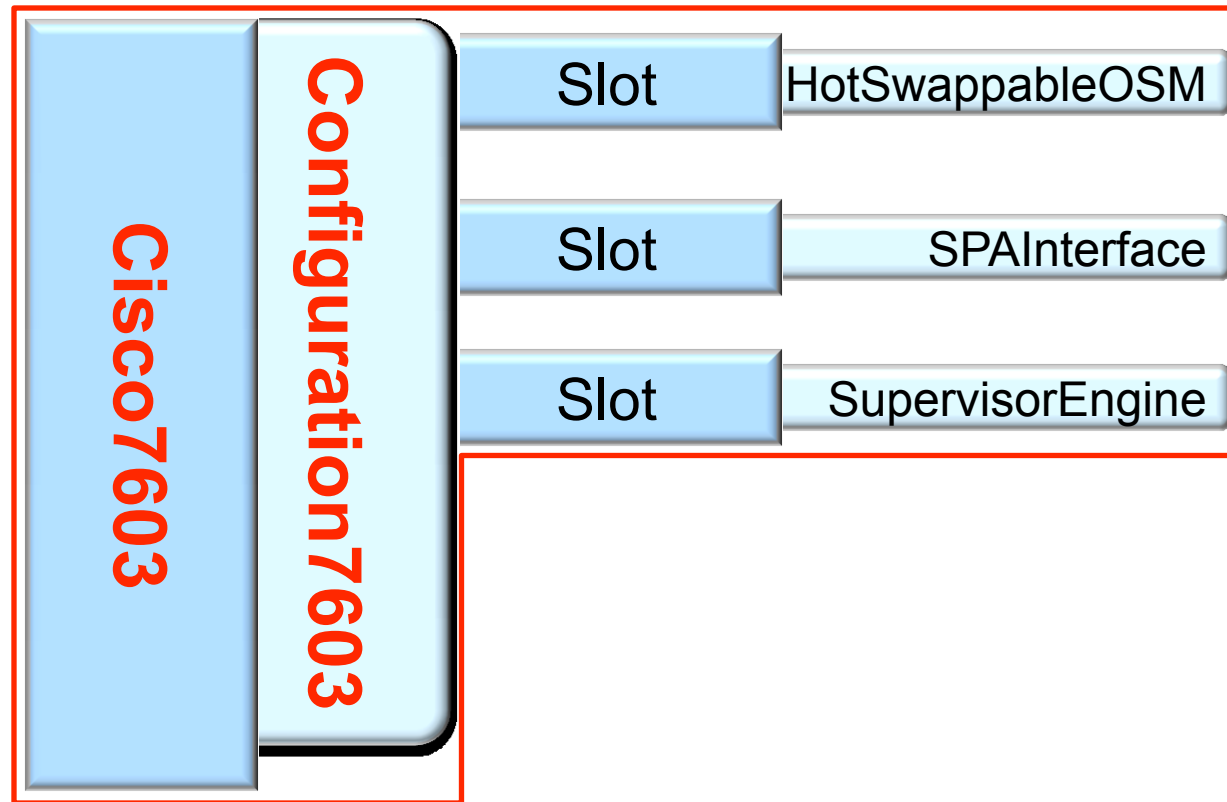
- Creates Metamodels to specify the Domain Specific Language
- Provides Concrete Syntax to DSL Users
- Supports the DSL User, e.g. by Guidance, Validation

- Modeling Physical Devices, e.g. Cisco Network Devices

Cisco 7603:



- Structure of a Device
  - Device -> Configuration -> Slot -> Card



- Requirements of DSL User:
  - Validation and Consistency Checking
  - Automatically Refinement of Domain Models

## Metamodel of Network Devices (in concrete Syntax)

- Implemented using KM3, a DSL to define Metamodels

```
class Device {
  reference hasConfiguration [1-*]: Configuration;
}

class Cisco7603 extends Device{
}

class Configuration {
  reference hasSlot [1-*]: Slot;
}

class Configuration7603 extends Configuration{
}

class Slot {
  reference hasCard [1-*]: Card;
}

class Card {
}
```

## Requirements for DSL Designer

- Define Constraints and Axioms
- Define Formal Semantics

E.g. DSL Designer wants to define:

- Every *Cisco7603* has at least 1 *Configuration7603*
- Every *Configuration* has at least 1 *Slot* in which a *SupervisorEngine* Card is plugged in
- A *Configuration7603* has exactly 3 *Slots* in which either a *HotSwappableOSM* or *SPAInterface* is plugged in.

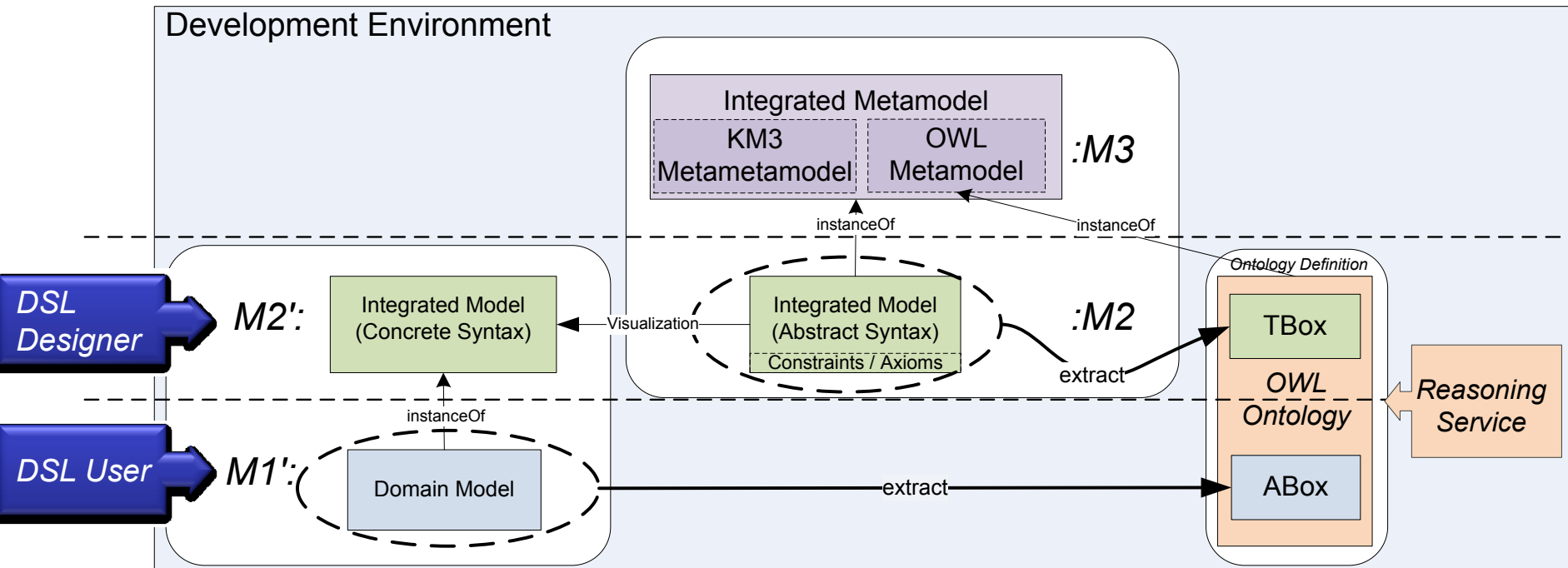
- Metamodel of Network Devices (in concrete Syntax)

```
class Device {  
  reference hasConfiguration [1-*]: Configuration;  
}  
  
class Cisco7603 extends Device, equivalentwith restrictionOn hasConfiguration  
with min 1 Configuration7603 {  
}  
  
class Configuration equivalentwith  
IntersectionOf(restrictionOn hasSlot with min 1 Slot,  
restrictionOn hasSlot some  
restrictionOn hasCard some SupervisorEngine) {  
  reference hasSlot : Slot;  
}  
  
class Configuration7603 extends Configuration,  
equivalentwith IntersectionOf(restrictionOn hasSlot with exactly 3 slot,  
restrictionOn hasSlot with some  
restrictionOn hasCard with some  
UnionOf(HotSwappableOSM, SPAInterface) {  
}  
  
class Slot {  
  reference hasCard [1-*]: Card;  
}
```

## Ontology-based Framework for Domain Specific Languages

- Integration of KM3 with an Ontology Language
  - Provide a Language to specify further DSLs
  - Used by the DSL Designer
- Designing Domain Specific Languages
  - Develop new DSL with integrated Constraints and Axioms
  - Provide the specified DSL to the DSL User
- Using Domain Specific Languages
  - DSL User gets Benefits of different Services

- DSL Designer
  - Abstract Syntax, Concrete Syntax, Semantics
- DSL User
  - Domain Models



## DSL Designer

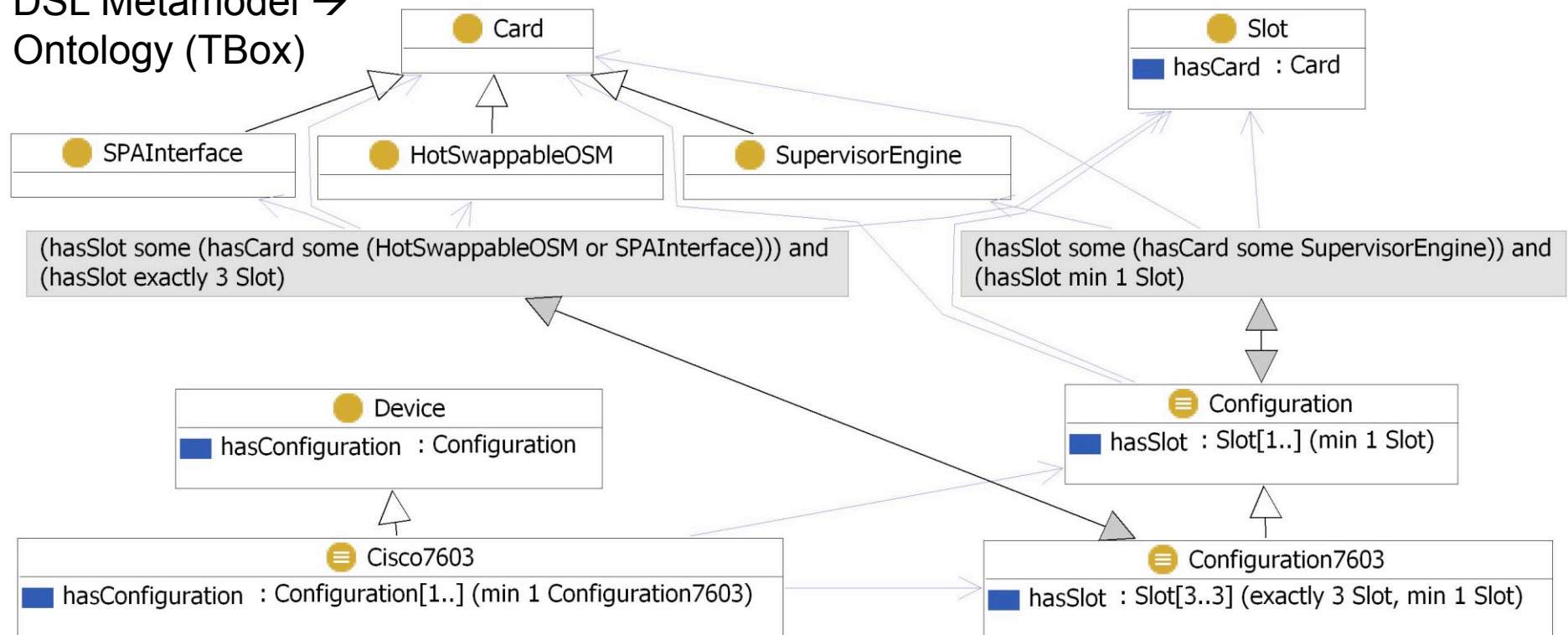
- Constraint Definition
- Language to define Formal Semantics of instances
- Checking Concept Satisfiability

## DSL User

- Services provided to the DSL User
  - Dynamic Classification
  - Consistency Checking
  - Debugging
- Services provided to the DSL User without any extra effort

- To adopt Reasoning Services on DSL Metamodels and Domain Models, both are transformed to one Ontology (TBox and ABox)
- Result: Reasoning simultaneously on M2- and M1 layer is provided

DSL Metamodel →  
Ontology (TBox)



## Accomplished Service:

- Determines the most specific type an Model Element belongs to
- With respect to all Attributes and Properties in the Context of the Model Element

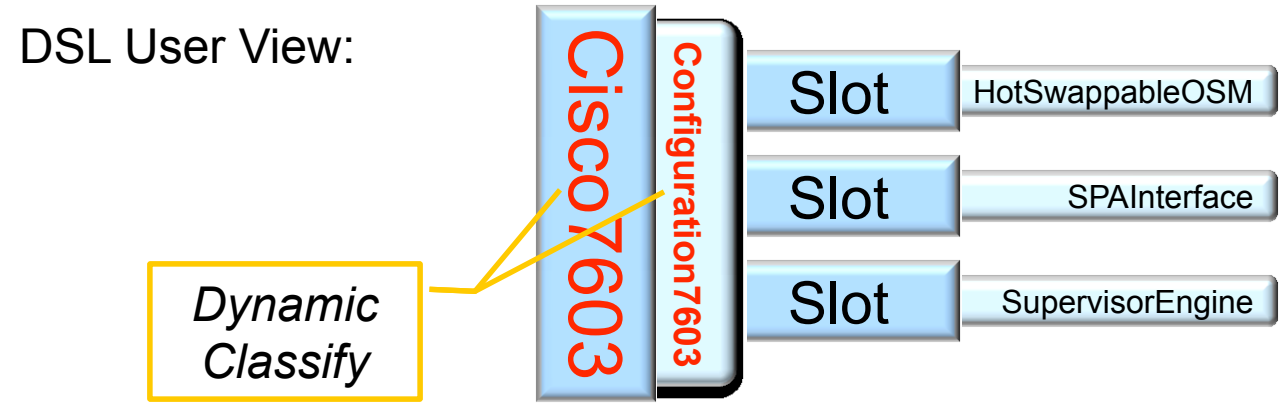
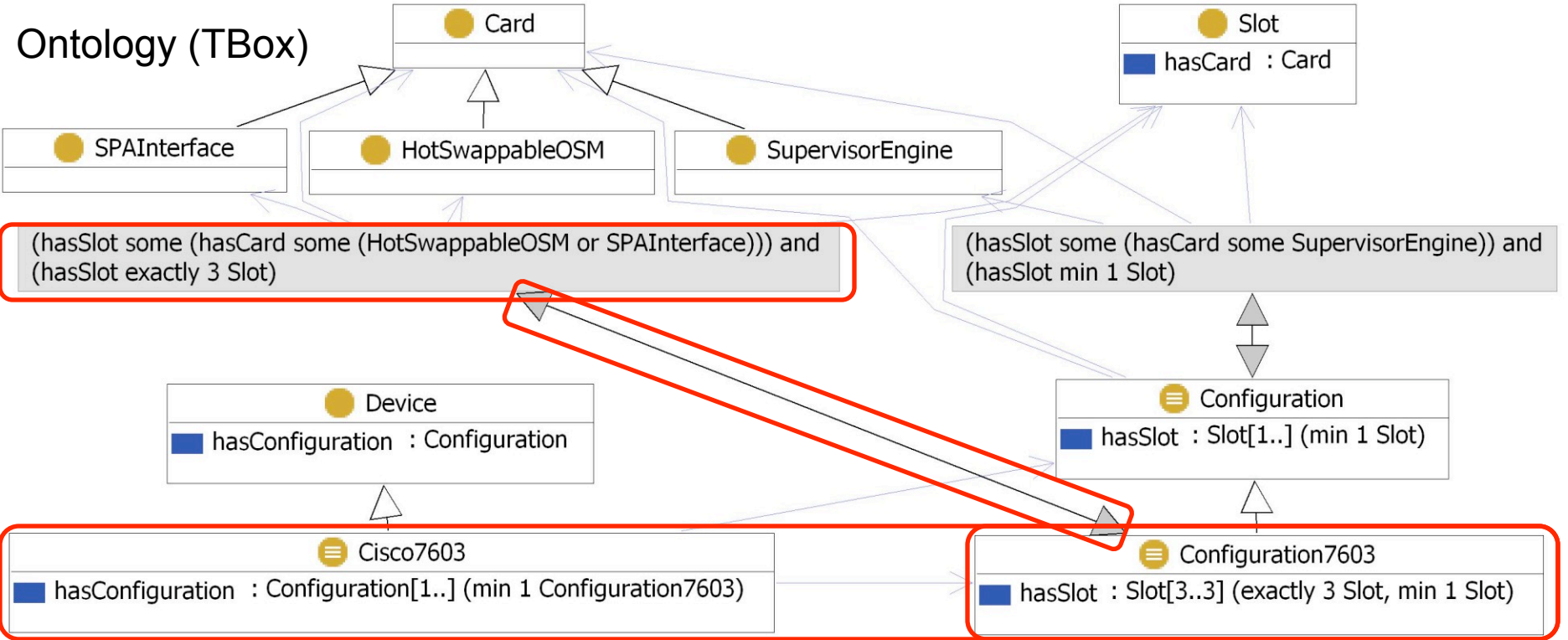
## Requirements for DSL Designers:

- Define Axioms
- Possibility to define Constraints and Restrictions

## Benefits for DSL Users:

- Automatically Refinement of Model Elements
- Suggestions of suitable domain concepts to be used

# Dynamic Classification (Example)



## Accomplished Service:

- Ensures that a Domain Model does not contain any contradictory facts with regard to its DSL Metamodel

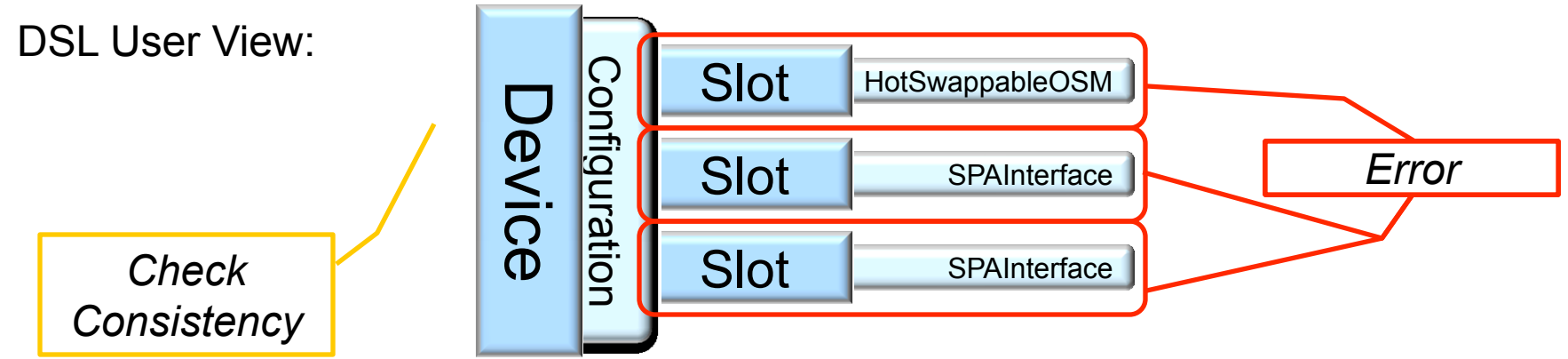
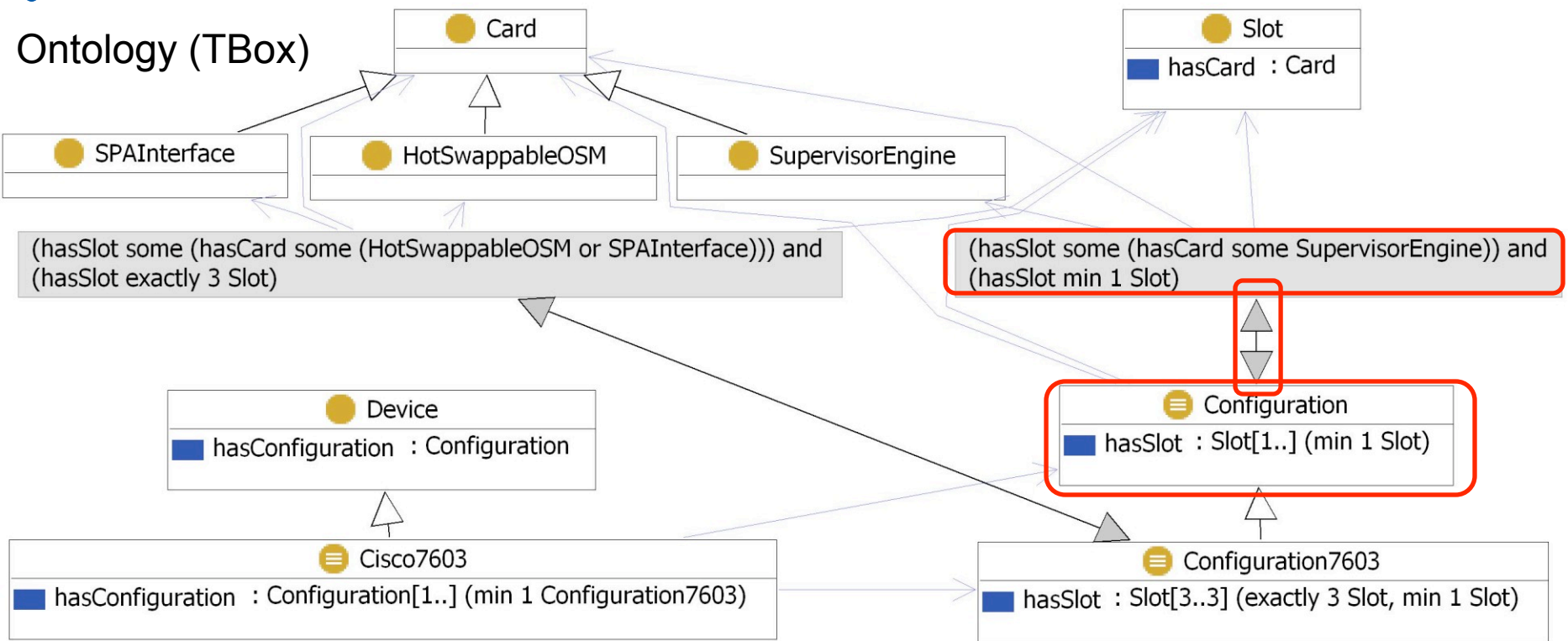
## Requirements for DSL Designers:

- Possibility to define Constraints and Restrictions
- Define Axioms

## Benefits for DSL Users:

- Qualitative Domain Models

# Consistency Checking (Example)



## Integration of KM3 and OWL

- Providing a Language that allows to use DSL and OWL Constructs simultaneously

## DSL Designer

- Specifies new DSLs with additional, integrated Constraints

## DSL User

- Builds Domain Models
- Reasoning Services

Thanks for your attention ...

... Questions?

... Comments?

... Ideas?

<http://ontodsl.semanticsoftware.eu>

Supported by EU STReP-216691 MOST